Stochastic and Asynchronous Spiking Dynamic Neural Fields

Benoît Chappet de Vangel, Cesar Torres-Huitzil, Bernard Girau

Stochastic and Asynchronous Spiking Dynamic Neural Fields

Benoît Chappet de Vangel
Université de Lorraine, France
Email: bdevangel@gmail.com

Cesar Torres-Huitzil
Cinvestav Tamaulipas, Mexico

Bernard Girau
Université de Lorraine, France

***Abstract*—Stochastic computing was extensively studied for artificial neural networks (ANN) implementation with a good time/area trade-off on up-to-date FPGAs. We propose here to use the same paradigm for the hardware implementation of dynamic neural fields (DNF) on FPGAs. The all-to-all connectivity of these neural population models make straight-forward hardware mappings impossible for high density fields. It is necessary to adapt the architecture to fit the cellular nature of computing substrates such as FPGAs. Following the previous work on randomly spiking dynamic neural fields, we propose here a new implementation inspired by stochastic ANNs. We introduce here the Cellular Array of Stochastic Asynchronous Spiking DNF model, or CASAS-DNF. While keeping the fully decentralized cellular characteristics, this new approach is much more competitive in terms of speed and area. We also show that the basic behaviors of DNFs are preserved. The low hardware cost and the cellular design of this model make it easily scalable.**

I. Introduction

Artificial neural networks (ANNs) are well known as a tool inspired from biology and applied successfully in several fields such as pattern recognition, language processing, classification, etc. Numerous hardware options have been developed to rise the processing power of these ANNs. Different implementations include analogue or digital hardware substrate [1]. Among digital hardware FPGAs are often retained as the best trade-off between logic density and design time.

The usual limiting factor of hardware implementation of ANN is the synaptic density. Usual ANN have 10 to 1000 time more synapses than neurons and their hardware implementation is costly as a synapse requires multiplying a neural activity by a specific synaptic weight. Several options were proposed to address this issue: time multiplexing, serial arithmetic or stochastic arithmetic.
Stochastic arithmetic (reviewed in [2]) describes a real value with a bit-stream where a bit has a probability p to be high and $1-p$ to be low. The advantages are numerous as the operators for complex operations are very light. The multiplication can be computed with a simple AND gate while the sum can be approximated by an OR gate (to some extent) or by a multiplexer. These principles have been successfully applied to FPGA-based ANN implementations (see [3], [4]).

We focus here on using the same principles to implement a widely used tool for bio-inspired cognition: the dynamic neural fields (DNF) introduced by Beurle [5] and Wilson [6]. The DNF is a mathematical tool built on the continuous neural field theory (CNFT), which models the layers of cortical columns in the brain as a continuous field. Many applications exist, especially for perceptual tasks and autonomous robotics [7].

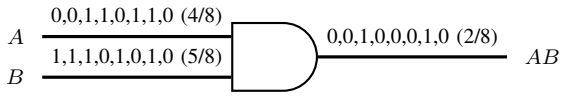
In [8], we introduced the randomly spiking dynamic neural fields (RSDNF). The idea was to adapt DNF to hardware in a cellular way. We used randomly propagated spikes in a regular mesh of von Neumann connected neurons so as the spike propagation and randomization would induce a virtually fully connected neural network with a Gaussian shaped lateral weights function. If functionally the random behavior was not altering the model abilities, the hardware performance of this decentralized model was not competitive compared to a centralized spike routing approach [9]. Besides this attempt, no dedicated hardware implementation has been proposed for dynamic neural fields. In this paper we propose a new approach based on stochastic computation. It maintains the idea of using a regular mesh but instead of propagating spikes randomly, stochastic bit streams are propagated taking advantage of the simplicity of their operators. This approach has resulted in the cellular array-based stochastic and asynchronous spiking dynamic neural field (CASAS-DNF) model that we describe in this paper.

II. Related work

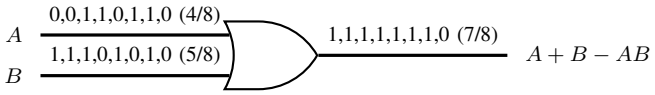
A. Hardware implementations of stochastic artificial neural networks

In the 90's, two works explained the potential of stochastic neurons (bit-stream neurons or pulse mode neuron) for ANN FPGA implementation, see [3] and [4]. But the area consumed by pseudo random number generators was limiting the applications at that time. With the rising power of FPGAs, stochastic ANNs were studied more recently by [10], [11] and [12].

The idea of stochastic computing is to represent numbers with the frequency of '1's in a bit-stream. For instance we can represent the number $v = 1/3$ with a bit stream of size $s = 9$ with 3 '1's and 6 '0's. More generally the bit stream of arbitrary size s and value v will be generated with a probability $p = v$. The definition can then be extended to encode values $v > 1$ (unipolar representation) and even negative values (bipolar representation). The main advantage of stochastic bit streams is that multiplication can be implemented with a single AND gate. Neuron activation functions can also be implemented with simple operators. The sum is less straight forward than the multiplication: the OR gate with two bit streams A and B gives a bit stream $A + B - AB$ (see fig 1). Another solution is to use a multiplexer which will return a bit stream encoding $(A + B)/2$. To encode numbers



(a) Stochastic Multiplication



(b) Stochastic Addition

Fig. 1. Bit stream operators. AND gate can approximate 4/8 x 6/8 while OR gate can approximate 4/8 + 5/8 with an error of -4/8 x 6/8. The results are not very precise as a stream of size 8 is shown in this example.

into stochastic bit streams, stochastic number generator (SNG) requires a pseudo random number generator (PRNG). The pseudo random numbers are then compared to a probability p and the result will generate a bit stream of probability p. The quality of the PRNG and the precision of the comparison with p are two parameters which will influence the precision of stochastic computation. The number of PRNG has to be scaled with the number of SNG as the random numbers correlation leads to critical drop of stochastic bit stream quality. The historical approach to produce pseudo random numbers used linear-feedback shift registers [13]. Their quality can be very good but their area cost has motivated research for new biomorphic cellular hardware (see [14] and [15]). To decode a stochastic number, the stochastic number decoder (SND) has to count the number of ones in a bit stream with a simple counter.

In summary the operators to manipulate stochastic bit-streams are very competitive in term of area except for the SNG which is heavier and critical for computation quality. The quality of the stochastic representation depends on the size of the bit-streams, the quality of the stochastic number generators, the quality of the operators and the bit stream correlation. This limits will have to be tuned carefully to optimize the speed-area trade-off.

B. Cellular computing and spiking neural networks

Despite their intrinsic parallelism and the simplicity of their computation, neural networks are not easy to map onto parallel hardware devices. With the rising computational power needed for precise simulations of neuro-biological models, a whole field of science, namely neuromorphic engineering [16], is trying to find new hardware paradigms for large-scale spiking neural networks simulation. The main problem for vast neuron dynamic simulation is the non-locality of the computation which is highly incompatible with state of the art computing devices. The spiking neural networks (SNN) address partly this issue by reducing the non-local informations size to 1 bit. Specific networks on chip (NoC) for SNN were developed to improve the speed of spike propagation based on the Address Event Representation paradigm (AER) [17]. The idea is to sequentially address every spike to its destinations using a dedicated high speed bus or a binary tree [18]. These solutions can be an inspiration for hardware artificial network implementations, and the speed-area trade-off is generally very competitive. In [9], we derived an AER-based hardware implementation of spiking DNFs with good performance. However, this sequential processing of spikes is not satisfying as it lacks the main characteristic of the brain: its decentralization. We are more interested in the advantages of decentralized cellular hardware for their numerous advantages [19]. It perfectly fits our current hardware paradigm with the local computation and good parallelism. It is fault tolerant as the loss of one element does not alter the general behavior. And finally, the computation is an emergent property of the system exactly like in the brain. This emergence confers a very competitive robustness to noise and is thus compatible with the introduction of stochastic computation.

Attempts to mix cellular computing and neural networks already exist: locality in hardware implementation of ANNs was achieved with FPNAs (in [20] and in [21] with up-to date boards), while cellular neural networks are already well-known [22] but they handle purely local communications between neurons.

III. From DNFs to hardware friendly DNFs

A. Dynamic neural fields

Dynamic neural fields (DNFs) were first introduced to model neural activity at a mesoscopic level of a cortical column. It is an approximation initially based on the very simple assumption of the continuum neural field theory (CNFT) introduced by Wilson [6] and developed by Amari [23]. In this theory, the neural population of a cortical column is approximated by a locally cooperative and globally competitive neural field. The neurons are rate-coded and their potential ($u_{x,t}$ for neuron x at time t) depends on the whole potential field M, following the equation (already discretized with an Euler method):

$$u_{x,t+dt} = u_{x,t} + \frac{dt}{\tau}(-u_{x,t} + \sum_{M} \omega(||x - x'||)\sigma(u_{x',t})dx' + I_{x,t} + h), \tag{1}$$

where dt is the discretizing step, $I_{x,t}$ the input feeding, h the resting potential, τ the time constant and ω is the lateral weights function. σ is the activation function which can be a sigmoid and ω models inhibitory and excitatory synaptic weights according distance d between neurons. This lateral weights function ω is a difference of Gaussians (DoG):

$$\omega(d) = Ae^{\frac{d^2}{a^2}} - Be^{\frac{d^2}{b^2}}, A, a, B, b \in \mathbb{R}_+^*. \tag{2}$$

The dynamic of the DNF (reviewed in [24]) is very rich and interesting. The characteristics we are interested in for visual processing applications are the selection and tracking abilities [25]. It relies on a trade-off between the afferent feeding (the inputs I) and the lateral feeding term $\sum_M \omega(||x - x'||)\sigma(u_{x',t})dx'$. The global competition within the field is the negative part of the lateral weights function (representing inhibitory connections), while the local cooperation is the positive part of ω (representing local excitatory connections).

Taken as it is, a straightforward hardware implementation of a DNF would be problematic as it would need a very costly convolution operator to compute the lateral feeding.

B. Spiking dynamic neural fields

The first step towards an easier hardware implementation was to introduce spiking neurons which would exchange only binary informations. Spiking dynamic neural fields have been introduced by [26] and developed by [27]. This model is based on Leaky Integrate-and-Fire (LIF) neurons which have the property to fire a spike and reset their potential when the potential reaches a threshold. The discretized version of the SDNF equation follows:

$$u_{x,t+dt} = u_{x,t} + \frac{dt}{\tau}(-u_{x,t} + I_{x,t} + h) + I^{syn}_{x,t}. \quad (3)$$

The lateral influence is computed by applying the synaptic weight (ω_M) of eq. 2 to each received spike(S):

$$I^{syn}_{x,t} = \sum_{x'} \omega(||x - x'||)\sigma(u_{x',t}). \quad (4)$$

Finally the neurons emit a spike when their potential reaches a threshold θ.

$$\sigma(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta. \end{cases} \quad (5)$$

If a spike is fired, the potential is reset: $u_{x,t} = h$.

The dynamics being harder to describe with analytical tools, more work need to be done to fully understand their behavior. However, basic abilities like selection, tracking and robustness were demonstrated to surpass DNF abilities [27].

C. Randomly spiking dynamic neural fields

Despite their reduced 1-bit bandwidth, the spiking dynamic neural fields are fully connected neural networks. Therefore their hardware implementation remains problematic as we need a personalized all-to-all connectivity in the network.

The first attempt to reduce neuron connectivity to 4 neighbors was introduced with the randomly spiking dynamic neural field model (RSDNF) [8]. To simulate global connectivity with local 4-neighbors connectivity, we need to be able to simulate as precisely as possible the lateral weights function ω. As the connections are local, information between neurons (in our case spikes) can only be transferred from cell to cell. This induces a propagation delay to wait for every information to reach its destination.

In RSDNFs, the idea is to simulate the DoG's shape by using probabilities. For that, every spike activation information is propagated from the activated neuron to the borders of the map with a probabilistic network on chip (NoC). A spike is subdivided in N packets of one bit, which are then sent through the network where every hop depends on a probability p. The NoC is designed such that there can only be one path between two neurons a and b and such that its distance (in number of hop) is the Manhattan (or XY) distance between the neurons: $dist_{XY}(a,b) = |a_x - b_x| + |a_y - b_y|$. There is one NoC for the excitatory weights (the probability of each hop is p_a) and another one for the inhibitory weights (the probability of each hop is p_b). Consequently the decentralized lateral weights function is, on average,

$$\omega(a,b) = \omega(d = dist_{XY}(a,b)) = Ap_e^d - Bp_i^d. \quad (6)$$

The shape of the corresponding curve is different from equation 1 (see Fig. 2), but new optimal parameters where derived from optimization methods (see [28] and [29]).

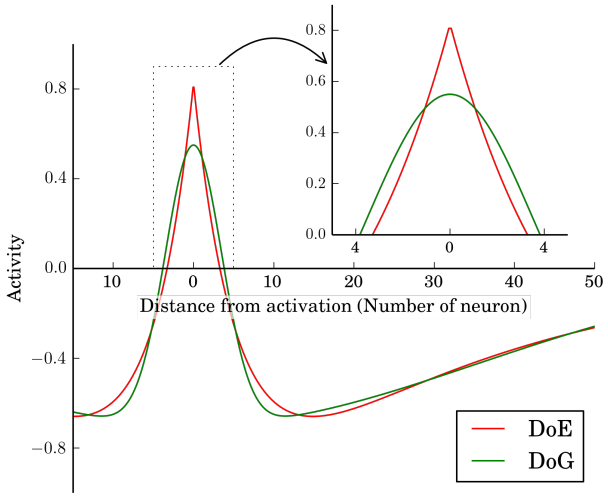


Fig. 2. Difference between the DNF lateral weights function which is a difference of Gaussian (DoG) and the RSDNF lateral weights function which is a difference of exponentials (DoE).

Behavioral simulation results confirmed the validity of the approach and the scalability of the cellular architecture of this model. However, the RSDNF speed×area product is not very competitive, and asynchronous neuron updates are not possible as a neuron has to wait for the end of spikes propagation to compute its next potential. This can be a problem to extend the RSDNF to synaptic delayed DNF [30], or to implement asynchronously computing DNFs [31].

IV. The CASAS-DNF model

As in the RSDNF paradigm, the hardware mesh is regular and composed of relatively simple identical cells. Each cell contains one neuron, two dendrites, eight synapses and eight routers (one inhibitory and one excitatory for each neighbor). The cells are disposed on a regular mesh connecting each other to their north, south, east and west neighbors.

A. Stochastic decentralized lateral weights

The challenge of cellular simulation for a globally connected neural networks is to have a good simulation of the synaptic weights between every neuron: in our case it is the lateral weights function ω. To assert the quality of the model we introduce in this work two criteria: the curve comparison (we compare our theoretic lateral weights function with the DNF one), and behavioral comparison (we compare our model dynamic with the standard model dynamic on a set of simulation scenarios). In this paragraph we present the analytical results for the shape of the CASAS-DNF lateral weights.

As we saw in eq. 2, the DNF lateral weights function ω only depends on the distance between neurons. Thus, following the idea introduced in RSDNF, we propose to simulate the

lateral weights using probabilities that decrease with distance, since it was proved to have a good curve similarity (Fig. 2) and a similar behavior despite the introduced randomness (see [8]).

We are using stochastic computation (or bit-stream arithmetic) to reproduce the lateral weights decreasing with the distance. The representation of a real number a is made by a bit-stream a_{bs} generated with a probability $a_p = a$, such that $P(a_{bs} = 1) = a_p = a$. To simplify the demonstration, we consider that the value of a bit stream is its probability.

Every activated neuron generates a bit-stream δ_{bs} of size s representing a spike of real value δ with a probability $\delta_p = \delta$. The spike will then be propagated throughout the network, being multiplied by synaptic weights at each hop. The synaptic weights do not depend on the neuron or synapse: the constant weight W_e is used for the excitatory layer, and W_i for the inhibitory layer.

To compute the lateral weights function for one spike we need to model the excitatory bit-stream received by the excitatory dendrites (E) and by inhibitory dendrites (I), assuming these dendrites belong to a neuron at a distance d from the activated neuron:

$$\begin{aligned} P(E_{bs} = 1) = E = \delta W_e^d \\ P(I_{bs} = 1) = I = \delta W_i^d \end{aligned} \tag{7}$$

Hence, the probabilistic lateral feeding equation integrated by the neuron is:

$$\omega_M(d) = I_e \delta W_e^d - I_i \delta W_i^d, \tag{8}$$

where I_e and I_i are respectively the intensity of excitatory spikes and inhibitory spikes, defined by:

$$I_e = \frac{A}{\delta} \text{ and } I_i = \frac{B}{\delta} \tag{9}$$

As this lateral weights equation is very similar to (eq. 6), we will use similar parameters. However, the bad quality of stochastic addition performed by the OR gate will bias these lateral weights.

The cellular nature of the network and the connectivity graph of the routers result in horizontal routers (east and west) having 4 predecessors, while vertical routers (north and south) have 2 predecessors (see fig. 4). Every router generates its outputs by summing the predecessors bit-streams, and as we saw in the stochastic computation paragraph, there is no perfect way to perform stochastic bit-stream addition. The reason is that as we sum numbers in the interval $[0, 1]$, the sum can be greater than the output bit-stream interval which is also $[0, 1]$.

Let A be an activation patch of n neurons (a set of n contiguous neurons that spike simultaneously). Let b represent one neuron at distance d from the patch (to simplify we assume that the neuron b is at distance d from every neuron of the patch). If we adapt the bit-stream integration in eq. 7 for the integration of n spike bit-streams instead of one, we have:

$$\begin{aligned} E = n\delta W_e^d \\ I = n\delta W_i^d \end{aligned} \tag{10}$$

It is important to note that the bit-stream additions are performed in several places: in the routers when several bit-streams are going through the same router and in the dendrite module as it receives the 4 neighbors bit-streams. Equation 10 is valid for a standard addition. However, as we are using OR gate for addition, we need to adapt it.

The generalized stochastic sum on n stochastic bit-streams of probability p using OR gate is

$$\sum_n p_{bs} = 1 - (1 - p)^n. \tag{11}$$

Consequently eq. 10 becomes

$$\begin{aligned} E = 1 - (1 - \delta W_e^d)^n \\ I = 1 - (1 - \delta W_i^d)^n \end{aligned} \tag{12}$$

Modifying eq. 8 to take into account the integration of n bit-streams using OR gates, the lateral influence of the patch of neurons A on neuron b becomes:

$$\omega_M(A, b) = I_e(1 - (1 - \delta W_e^d)^n) - I_i(1 - (1 - \delta W_i^d)^n). \tag{13}$$

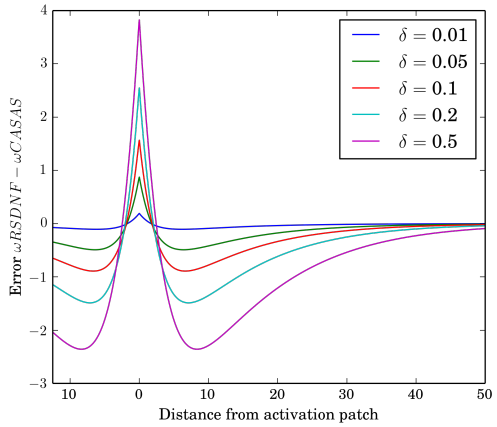
This is an approximation as we assumed that all the neurons of the patch A were at the same distance from neuron b and we did not take into account the precision of the bit-streams (their length s) which influences the quality of encoding and decoding. But it is enough to predict the influence of the different parameters on the lateral weights quality of the model. On Fig. 3 we see that the error rises with δ and n. When δ is greater than 0.01 and n is greater than 5 the error with the RSDNF lateral weights function is too important to be neglected. Consequently a new set of parameter has to be found. In this work we used particle swarm optimization techniques to find optimal parameters for the set of task we are presenting in the results section.

B. Spiking neuron

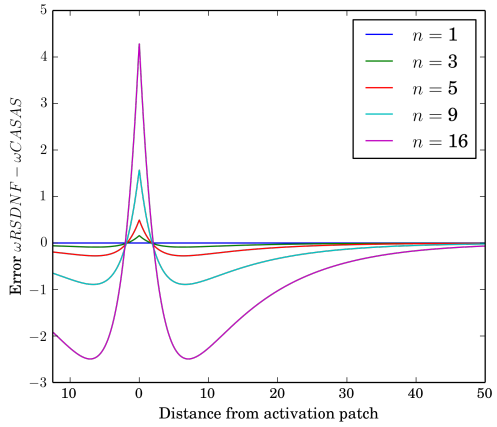
The dendrite module is a stochastic number decoder (SND). This modules receives the 4 neighbors bit-streams, sum them with an OR gate and add a quantum of excitation (q_e) or a quantum of inhibition (q_i) to a register when the output of the OR gate is high ($q_e = I_e/s$ and $q_i = I_i/s$) . When the neuron computation is activated, this lateral influence is integrated in the neuron potential, and the dendrite register is reset. The spiking neuron computation was made as close to the spiking neuron definition as possible (see Fig. IV-B). The only modification was made to transform the leaking factor division by a right bit shift to avoid a division. Details are available in [8].

C. Pseudo random numbers generation

The stochastic number generator module (SNG) uses pseudo random numbers (PRNs). We have shown in [8] that an 8-bit precision is enough for synaptic weight probabilities. However, the efficient generation of pseudo random numbers is often a complex question. Standard implementations use linear feedback shift registers but we are using massively replicated cellular automata which are a scalable and distributed way to generate multiple high quality PRNs [32]. To be fair with



(a)



(b)

Fig. 3. The error between the RSDNF lateral weights function ($\omega RSDNF$) and the CASAS_DNF lateral weights function ($\omega CASAS$) as a function of the neural distance from the activation patch (in number of neurons). Fig. 3a show the error with a rising value for spike probability δ and a fixed number of activated neurons $n = 9$ and Fig. 3b show the error with a rising number of activated neuron n and with $\delta = 0.1$.

RSDNFs, we never use twice a random bit, but it can be noted that a high random numbers re-utilization is compatible with our simple applications. For this architecture we need to generate four random numbers per synapse and per propagation cycle and one random number per neuron for the spike bit-stream generation. For a network of 35x35 neurons with excitatory and inhibitory synapses, it results in 11025 random numbers with 8 bits minimum. These 88200 random bits are simultaneously generated by Cellular Automata-based Pseudo-Random Number Generators (CAPRNG) that are very well adapted to FPGA devices and to our cellular computation requirement: non-homogeneous cellular automata with high random quality emerging from simple logic rules:

$$\sigma^{t+1}_{(i,j)} = X \oplus C.\sigma^{t}_{(i,j)} \oplus N.\sigma^{t}_{(i-1,j)} \oplus W\sigma^{t}_{(i,j-1)} \oplus S.\sigma^{t}_{(i+1,j)} \oplus E\sigma^{t}_{(i,j+1)} \quad (14)$$

where σ^t_c is the state of the cell c at time t, and where

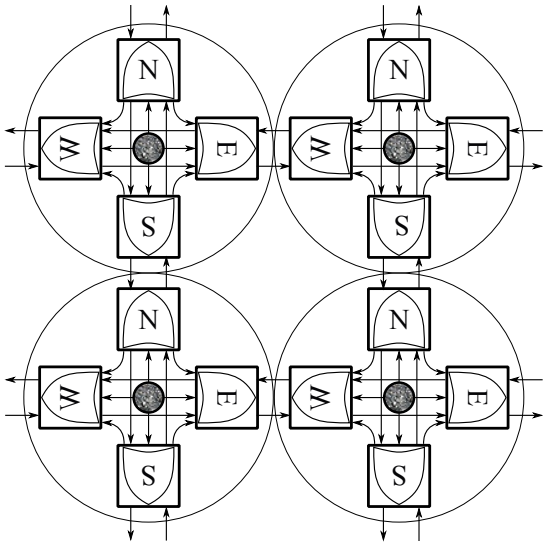


Fig. 4. Design of the routers and their connectivity. The CASAS cells (white circles) are organized on a 2D-grid. Every router's direction (square) contains one OR gate performing a sum approximation of the inputs stochastic bit-streams. The vertical OR get input from the opposite direction neighbor while horizontal OR get input from opposite direction and vertical directions neighbors. Every direction receive as well an input from the spike's stochastic bit stream generated by the CASAS Cell's neural activation (gray circle). The CASAS map contains two layer of this routers: one for the excitatory bit streams and one for the inhibitory bit streams.

TABLE I. SIMULATION PARAMETERS FOR BEHAVIORAL COMPARISON BETWEEN RSDNF AND CASAS-DNF MODELS

Model	A	B	p_e	p_i	s
CASAS-DNF	1.16	0.87	0.035	0.99	100
RSDNF	1.25	0.70	0.004	0.90	100

$XCNWSE \in [0,1]^6$ are the control variables determining the rule of each cell in the cellular automaton. In [33], [34], genetic algorithm were used to find the best rules for 8x8 CA grid. This PRNG shows very good behavior in standard tests while offering an outstanding compacity for FPGA implementation.

V. RESULTS

The CASAS-DNF model was implemented in software and hardware to study more precisely its dynamics and performance. Software simulation was programmed with a Python/C++ framework simulating the cellular computation of CASAS-DNF. The simulation parameters are shown in Table I.

A. Behavioral results

Behavioral software simulations of the two models SDNF and RSDNF have already been reported. In [27] a behavioral study shows the properties of spiking DNF for basic visual attention tasks. In [8], the random behavior of the RSDNF is compared to the behavior of spiking DNF to assert the validity of the cellular approach in basic visual tasks. A short summary of the main results is presented here.

We consider three different scenarios to test the error during basic visual selective attention and tracking tasks. The error is measured with the euclidean distance between the barycenter

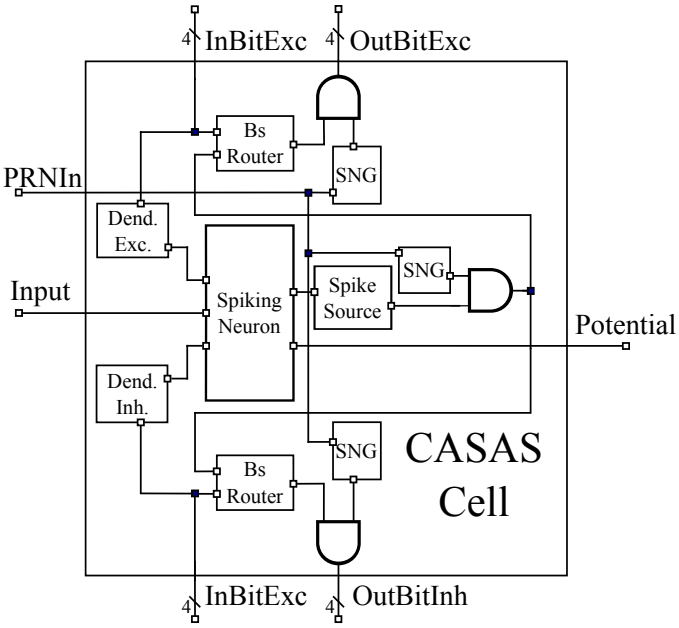


Fig. 5. Design of the CASAS cell. The 4 neighbors' bit stream and the cell's spike bit stream are routed by the bit stream router (Bs Router) before being multiplied by the synaptic weight generated using a stochastic number generator (SNG). They will then be transmitted to their destination neighbors. The input bit streams will also feed a so called dendrite module (Dent. Exc and Dent Inh.) which will simply sum (with an OR gate) decode and convert the bitstreams and send it to the neuron. The spiking neuron module compute the potential of the neuron. When the potential reaches a threshold it activates the activation module which will stay activated during s cycles. A AND gate will mutliply this activation signal with a stochastic number of probability δ. Note that every SNG modules are fed with pseudo random numbers (PRNIn) generated with a cellular automata.

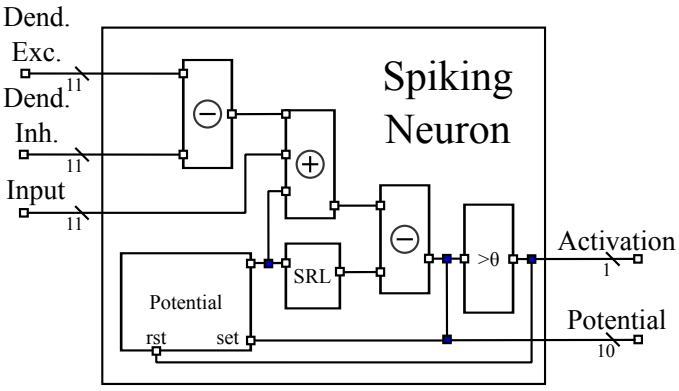


Fig. 6. Leaky integrate and fire neuron design. The input lateral excitation and inhibition are computed and added to the previous potential value. The leak is applied to the old potential with a division by 8 using a shift right logical (SRL) of 3 bits. Once the leak has been removed from the potential, the new potential is compared with the activation threshold δ. If the neuron is activated, the potential will be reset. Otherwise, the new potential is output and will update the potentials register.

of the field's activated neuron coordinates and the stimulus the field is focusing at. For instance we can see on Figure 7 the tracking scenario and the measured error. After nine computation iterations the neural field is spiking close to one stimulus: this is the selective attention or focus. When the stimulus circles, the activation of the field follows the stimulus.

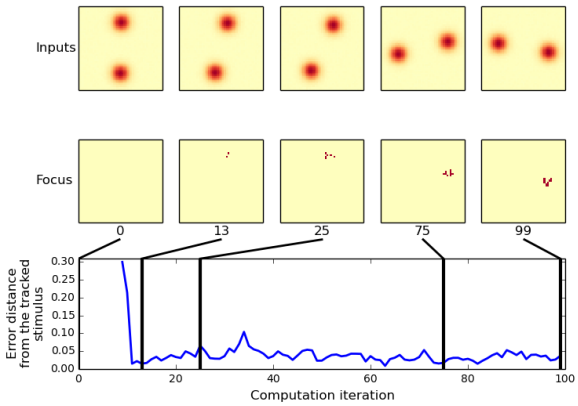


Fig. 7. Example of the competition scenario. The Inputs maps consist of two Gaussian bubbles rotating clock-wise. The Focus map which represents the neural field activations, tracks one bubble. The error distance between the barycenter of the field activation and the tracked bubble center is displayed on the bottom.

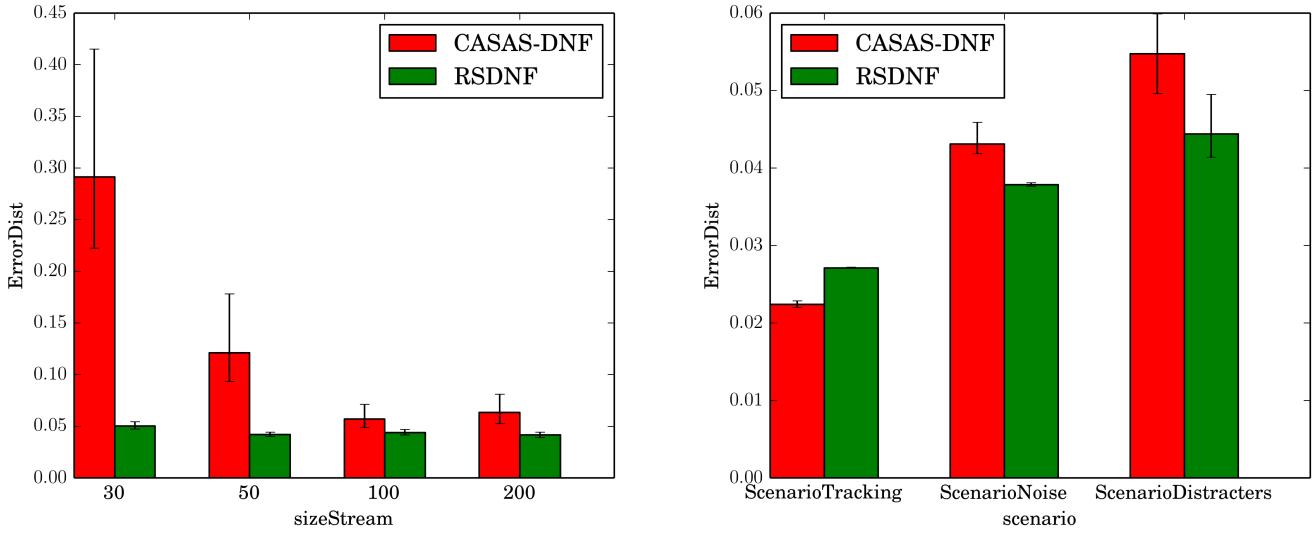
Figure 8a shows 50 iterations of tracking scenario for different sizes of the bit-stream s for CASAS-DNF and RSDNF. We can see that for CASAS-DNF s is optimal around 100. For Fig. 8b, we chose $s = 100$ and we repeated 50 experiments on the three different scenarios. The error bars show the bootstrap 95% confidence intervals. The model seems to be less tolerant to noise and distracters than RSDNF. It can be explained by the OR summation sensibility to the number of activated neurons n (see previous section discussions on lateral weights function quality).

B. Hardware utilization

The synthesis and placement was done on a FPGA Xilinx xc6vlx760t-3ff1156, the results on Fig. 9 show that the speed$\times$area product is much better and scalable for the CASAS-DNF model. The speed is not only computed with the maximum clock frequency of the design but also taking into account the computation speed of the model. All these models need time to propagate the spikes information throughout the neural map. This time is expressed in terms of propagation iterations. For CASAS-DNF, the propagation time lasts $s + 2res$ as a bit-stream of size s needs s iterations to be fully generated, and needs (at worse) $2res$ iteration to go through the whole map ($2res$ is the maximum Manhattan distance for our XY propagation with a resolution res). The update rate is computed with the division of the maximal frequency by the simulated propagation time and the SpeedxArea product is then computed with the division of the number of slice by the update rate.

We are comparing our architecture to RSDNF and to a completely centralized architecture based on the address event representation (AER) that we introduced in [9].

We can see on Table II that while CASAS-DNF has a slightly slower propagation time than RSDNF, it takes much less hardware resources. However the centralized AER-based implementation is still more competitive in terms of speed and area, though future CASAS-DNF optimizations (by partially sharing PRNs for example) might reverse this conclusion.



(a) (b)

Fig. 8. 50 repetitions of simulation for RSDNF and CASAS-DNF. On 8b we compare RSDNF and CASAS-DNF behaviors for 3 different scenarios. On 8a we show the distance error for rising qualities of stochastic bit-streams.

TABLE II. RESOURCE UTILIZATION AND PERFORMANCE OF THE ARCHITECTURE ON A XILINX FPGA DEVICE, XC6VLX760T-3FF1156, FOR 35X35 NEURONS NEURAL MAP.

Model	Registers	LUTs	Slices	Freq (Mhz)
CASAS-DNF	90497	112207	31851	127
RSDNF	233155	261011	86106	95.7
AER	74870	121266	46543	54.4

	Prop. it. theory and simu.		Update Rate (Mhz)	SpeedxArea
CASAS-DNF	$O(s+2res)$	170	0.36	42635
RSDNF	$O(nN+2res)$	120	0.7975	107970
AER	n	10	5.44	8556

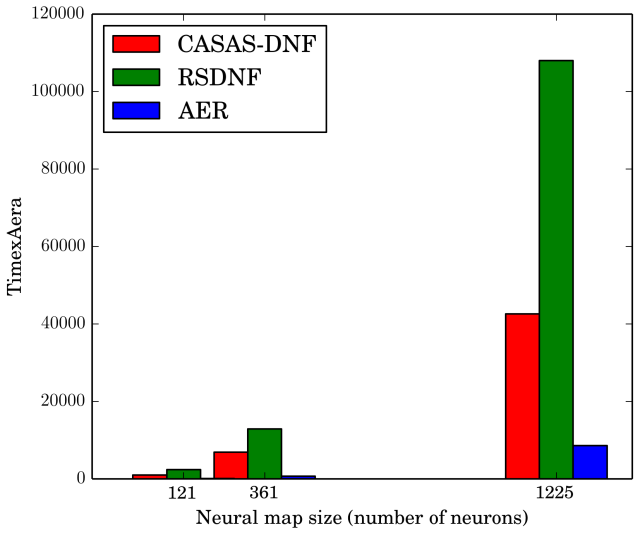


Fig. 9. Time×area performance for the different models and for different neural map sizes.

VI. CONCLUSION

We presented in this work a new DNF model with the main aim of facilitating its FPGA implementation. Using principles from stochastic computing, the CASAS-DNF model is a cellular array substratum which is able to simulate an all-to-all connectivity between thousands of neurons with a hardware friendly 4-neighbors connectivity.

Neuronal computations are by nature very robust. Stochastic bit stream operators are thus perfectly adapted as their intrinsic noise will not disturb the emergent behavior of the dynamic neural field. Therefore it is not surprising that they represent a very competitive speed-area trade-off for cellular computation of a globally connected neural network. It is also expected that precision and quality of the random numbers are not critical to the behavior. To be fair with the previous implementations, we used very good random numbers quality (one 8bit pseudo random number generation by propagation cycle and by cell routing layer direction: 88200 bits per cycle). But first simulations show that this number can be dramatically reduced. It seems that 5 bits precision is enough and that using the same 5 bits for every SNG of each CASAS cell is enough to maintain an identical behavior. Thus only 6125 pseudo random bits by cycle of bit propagation would be enough reducing the amount of CA-PRNG from a 37x38 grid to a 10x10 grid of 64 cells cellular automaton. Thus if the CA-PRNG are taking now more than 60% of the area a decreasing of the random number quality would reduce the overall area by 50%.

Beyond the improvement for hardware resources utilization compared to the RSDNF model, this architecture is adapted to much more DNF extensions than RSDNF. In this architecture the lateral-weights bit propagation is symmetric in every direction. It means that it will be possible to update neuron potentials on every bit reception instead of waiting the whole lateral-weights propagation time before updating

the neurons. With this architecture the pipelined simulation of DNF with synaptic delays will be possible, inducing a highly satisfactory time×area product. Future work will also focus on stronger predictions for lateral-weights quality and on the implementation of learning abilities.

REFERENCES

[1] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 13, pp. 239 – 255, 2010, artificial Brains. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523121000216X

[2] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 92:1–92:19, May 2013. [Online]. Available: http://doi.acm.org/10.1145/2465787.2465794

[3] M. van Daalen, P. Jeavons, and J. Shawe-Taylor, "A stochastic neural architecture that exploits dynamically reconfigurable fpgas," in *FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on*, Apr 1993, pp. 202–211.

[4] S. Bade and B. Hutchings, "Fpga-based stochastic neural networks-implementation," in *FPGAs for Custom Computing Machines, 1994. Proceedings. IEEE Workshop on*, Apr 1994, pp. 189–198.

[5] R. L. Beurle, "Properties of a mass of cells capable of regenerating pulses," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 240, no. 669, pp. 55–94, 1956.

[6] H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophysical Journal*, vol. 12, no. 1, pp. 1 – 24, 1972. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0006349572860685

[7] W. Erlhagen and E. Bicho, "The dynamic neural field approach to cognitive robotics." *J Neural Eng*, vol. 3, no. 3, pp. R36–54, 2006.

[8] B. Chappet de Vangel, C. Torres-Huitzil, and B. Girau, "Randomly spiking dynamic neural fields," *ACM Journal of Emerging Technologies in Computing Systems*, vol. 11, no. 4, pp. 1–26, Apr 2015.

[9] ——, "Spiking dynamic neural fields architectures on fpga," in *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, Dec 2014, pp. 1–6.

[10] H. Hikawa, "A digital hardware pulse-mode neuron with piecewise linear activation function," *Neural Networks, IEEE Transactions on*, vol. 14, no. 5, pp. 1028–1037, Sept 2003.

[11] N. Nedjah and L. Mourelle, "Fpga-based hardware architecture for neural networks: binary radix vs. stochastic," in *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, Sept 2003, pp. 111–116.

[12] J. Rossello, V. Canals, and A. Morro, "Hardware implementation of stochastic-based neural networks," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, July 2010, pp. 1–4.

[13] S. W. Golomb, L. R. Welch, R. M. Goldstein, and A. W. Hales, *Shift register sequences*. Aegean Park Press Laguna Hills, CA, 1982, vol. 78.

[14] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Bio-inspired stochastic computing using binary cbram synapses," *Electron Devices, IEEE Transactions on*, vol. 60, no. 7, pp. 2402–2409, July 2013.

[15] O. Kavehei and E. Skafidas, "Highly scalable neuromorphic hardware with 1-bit stochastic nano-synapses," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, June 2014, pp. 1648–1651.

[16] G. Indiveri and T. K. Horiuchi, "Frontiers in neuromorphic engineering," *Frontiers in Neuroscience*, vol. 5, 2011.

[17] M. Mahowald, "Vlsi analogs of neuronal visual processing: A synthesis of form and function," Ph.D. dissertation, California Institute of Technology, 1992.

[18] D. Vainbrand and R. Ginosar, "Scalable network-on-chip architecture for configurable neural networks," *Microprocessors and Microsystems*, vol. 35, no. 2, pp. 152–166, 2011.

[19] M. Sipper, "The emergence of cellular computing," *Computer*, vol. 32, no. 7, pp. 18–26, 1999.

[20] B. Girau, "Neural networks on FPGAs : a survey," in *Second ICSC Symposium on Neural Computation - NC'2000, Berlin, Germany*, 2000.

[21] M. Bohrn, L. Fujcik, and R. Vrba, "Field programmable neural array for feed-forward neural networks," in *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, July 2013, pp. 727–731.

[22] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *Circuits and Systems, IEEE Transactions on*, vol. 35, no. 10, pp. 1273–1290, 1988.

[23] S.-i. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biological Cybernetics*, vol. 27, pp. 77–87, 1977, 10.1007/BF00337259. [Online]. Available: http://dx.doi.org/10.1007/BF00337259

[24] S. Coombes, "Waves, bumps, and patterns in neural field theories," *Biological Cybernetics*, vol. 93, no. 2, pp. 91–108, 2005. [Online]. Available: http://dx.doi.org/10.1007/s00422-005-0574-y

[25] N. P. Rougier and J. Vitay, "Emergence of attention within a neural population," *Neural Networks*, vol. 19, no. 5, pp. 573 – 581, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S089360800500081X

[26] S. Chevallier and P. Tarroux, "Visual focus with spiking neurons," in *European Symposium on Artificial Neural Networks*, M. Verleysen, Ed. Bruges, Belgique: d-side, Apr. 2008, pp. 385–389. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00472642

[27] R. Vazquez, B. Girau, and J. Quinton, "Visual attention using spiking neural maps," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 2164 –2171.

[28] J.-C. Quinton, "Exploring and Optimizing Dynamic Neural Fields Parameters Using Genetic Algorithms," in *IEEE World Congress on Computational Intelligence 2010 - WCCI 2010*, Barcelona, Espagne, Jul. 2010. [Online]. Available: http://hal.inria.fr/inria-00488914

[29] J. Fix, "Template based black-box optimization of dynamic neural fields," *Neural Networks*, vol. 46, no. 0, pp. 40 – 49, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S089360801300124X

[30] A. Hutt and N. P. Rougier, "Activity spread and breathers induced by finite transmission speeds in two-dimensional neural fields," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 82, p. R055701, 2010.

[31] W. Taouali, T. Viville, N. P. Rougier, and F. Alexandre, "No clock to rule them all," *Journal of Physiology-Paris*, vol. 105, no. 13, pp. 83 – 90, 2011, computational Neuroscience: Neurocomp 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092842571100026X

[32] B. Girau and N. Vlassopoulos, "Tiled cellular automata for area-efficient distributed random number generators," in *1st International conference on pervasive and embedded computing and communication systems - PECCS 2011*, 2011.

[33] ——, "Evolution of 2-dimensional cellular automata as pseudo-random number generators," in *Cellular Automata*, ser. Lecture Notes in Computer Science, G. Sirakoulis and S. Bandini, Eds. Springer Berlin Heidelberg, 2012, vol. 7495, pp. 611–622. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33350-7_63

[34] N. Vlassopoulos and B. Girau, "A Metric for Evolving 2-D Cellular Automata As Pseudo-Random Number Generators," *Journal of Cellular Automata*, 2014.